所属类别	2020 年“华数杯”全国大学生数学建模竞赛	参赛编号

电动汽车销售策略的研究

摘要

随着不可再生能源的稀缺问题日益凸显，以电动汽车为代表的新能源产品逐渐受到人们的欢迎。电动汽车销售市场存在着巨大潜能，对意向购买电动汽车的客户进行研究，有助于电动汽车商家制定精准的营销策略，从而获取丰富的商业利润。本文基于三款品牌电动汽车，分别分析了其目标客户对汽车的体验数据和客户的个人特征，并针对数据分析结果给出了汽车销售策略。

对于问题一，本文首先通过绘制箱线图，挖掘出落在四分位数加减 1.5 倍四分位距之外的潜在异常值。接下来研究了存在少量缺失值的特征 B7 与 B6 之间的关系，并对缺失值进行了填补。最后，绘制不同品牌服务满意度小提琴图，并依据 Wilcox 统计量和 Kruskal-Wallis 统计量，比较了不同汽车品牌满意度之间的差异。

对于问题二，在对数值特征进行最大最小归一化基础上，使用 ADASYN 采样将数据转化为平衡的样本。然后，对于每一种汽车品牌，根据卡方检验的得分阈值，初步筛选出 4 个分类变量；使用最大信息系数法，依据最大信息系数二次筛选出 2 个最优的分类变量。针对数值型变量，使用方差过滤法，一次筛选出大于方差阈值的 13 个候选特征；使用随机森林模型，根据 Gini 指数二次筛选出重要程度前 8 的特征；使用 RBF 递归特征消除的方法，通过循环递归得到了最终的 10 个数值型特征。最终筛选出的特征即是影响电动汽车销售的因素。

对于问题三，首先对分类变量进行 One-Hot 编码，然后基于不同汽车品牌，各建立了四个分类模型：XGboost、LightGBM、Catboost 和随机森林，并使用贝叶斯调参对模型参数进行了优化。在此基础上，通过 AUC 值和 F1 值对模型进行了评估，得到模型的 AUC 值均大于 0.89，F1 值均大于 0.7，表明模型效果较好。最后，本文选出效果最好的模型对附录 3 中的客户购买行为进行了预测。

对于问题四，本文首先基于问题三中不同品牌目标客户挖掘模型，在满意度提升为整数的假定下，枚举出所有满意度提升百分点的组合，然后考虑到服务难度与提高的满意度百分点呈正比关系，提出服务难度最小的目标函数，建立提升满意度优化模型，通过网格搜索的思想，对目标客户（编号 2,8,15），求解出最优的提升满意度百分点组合，最后结合该组合对不同目标客户制定销售策略。

对于问题五，本文依据购买电动汽车的重要影响因素和前一问中得出的提高满意度的方向，对不同品牌电动汽车的营销建议涵盖了以下几点：

1. 销售人员要详细向客户介绍电池的优良性能，如电池耐用和充电方便等性能；
2. 销售部门可以通过广告、互联网等途径宣传汽车品牌的安全性和动力优越性，并推出车险赠送服务；
3. 推出客户免费试驾体验服务，赠送提高客户舒适度的相关汽车附属产品。

关键词：电动汽车销售；特征工程；XGboost；LightGBM；组合优化

一. 问题重述

1.1 问题背景

环境污染和温室效应等问题伴随全球化的发展日益严峻，提倡低碳生活和可持续化发展也越来越为国家所重视。其中，新能源汽车行业作为战略性新兴产业，得到了大力的发展，而新能源电动车因为其优越性更是为人们所关注[1]。一方面，石油的稀缺导致了石油价格的整体上涨，长此以往，石油这类非可持续资源必有耗尽的一天，因此发展新能源产业成了必然的趋势。另一方面，新能源在汽车行业的应用使得人们在面对高额的邮件时有了更多的选择[2]。虽然，新能源电动车有着许多优点，但由于其前沿性，人们会因为对他不够了解而产生一些顾虑。所以，了解目标用户对新能源电动车的顾虑以及满意点是我们销售新能源电动车的重点。

1.2 需要解决的问题

问题一：在我们已有的数据中，出现了数据缺失和数据异常的情况，我们需要用统计方法对数据进行清洗。并对数据进行描述性统计分析，同时还需要分析比较客户对不同品牌汽车的满意度

问题二：影响客户购买电动车的因素很多，如电动汽车自身的状况，也有目标客户个人情况。在本次目标客户体验的时候，有部分目标客户购买了体验的电动汽车（购买了用 1 表示，没有购买用 0 表示）。通过给定的信息，判断哪些因素会对不同品牌电动汽车的销售造成影响？

问题三：在之前的研究基础上，再建立不同品牌电动汽车的客户挖掘模型，对该模型的好坏进行评价。并运用模型判断附件 3 中 15 名目标客户购买电动车的可能性。

问题四：销售部门认为，满意度是目标客户汽车体验的一种感觉，只要营销者加大服务力度，在短的时间内提高 a1-a8 五个百分点的满意度是有可能的，但服务难度与提高的满意度百分点是成正比的，即提高体验满意度 5%的服务难度是提高体验满意度 1%服务难度的 5 倍。基于这种思路和前面的研究成果，请你在附件 3 每个品牌中各挑选 1 名没有购买电动汽车的目标客户，实施销售策略。

问题五：结合之前的研究，给销售部门提出不超过 500 字的销售策略建议。

二. 问题分析

2.1 问题一分析

通过观察发现 B8 特征不够直观，对数据进行清洗，得到 B8 新的特征。在做异常值处理时，一方面选取对数据分布没有要求的箱线图，对数值型数据进行检测；另一方面，通过观察规律，发现部分数据是人为失误所致，对其进行更改和删除处理。缺失值主要集中在 B7，通过发现 B6 和 B7 数据之间的关系以及中位数替代的方法，对缺失值进行替代处理。再用小提琴图和热力图对数据进行描述性统计分析，最后引入 Wilcox 统计量和 Kruskal-Wallis 统计量分析客户对不同品牌汽车的满意度。

2.2 问题二分析

为了寻找出影响电动汽车销售的因素，需要对预处理后的数据进行特征筛选。考虑到数据集中包含了数值型特征、分类型特征和有序特征，本文可以分别对数值特征和分类特征进行筛选，从而找出数据质量好、显著影响客户汽车购买行为的一些特征。

2.3 问题三分析

前一问中已经筛选出的影响客户汽车购买行为的因素，这些因素可以作为第三问中建模的特征。考虑到客户是否购买汽车是一个二分类变量，故可以构建分类器。由于分类模型种类繁多，因此可以考虑建立多个学习效果较好的集成学习模型，在此基础上进行比较和评估，选择其中最优的模型进行预测。

2.4 问题四分析

问题四要求在上述问题建模的基础上，对各项整体满意度做不同的提升，使得提升后的客户能够产生购买意愿，并找到使得满意度提升的同时，服务难度最小的提升组合。本文基于问题三建立的目标客户挖掘模型，在提升满意度百分点为整数的假定下，遍历所有可能的满意度提升组合，并结合附件三中的待测样本，采用网格搜索法的思想，得到所有使得待测目标客户愿意购买的提升满意度组合。进而结合提升满意度数据的内在联系，提出多种约束条件，建立基于提升满意度的规划模型，从而对于不同品牌的目标客户，找到服务难度最低的提升满意度组合，并对此制定销售策略。

三. 模型假设

1.假设异常值和缺失值处理后的数据有效可信，能够反映电动汽车与客户自身的因素；
2.假设除了本文提到的因素，没有其它因素影响到客户购买电动车的意愿；
3.假设各个因素之间是相互独立的，如 a1 因素不会对 a2 因素造成影响；
4.假设问卷得到的目标客户个人特征信息是真实可信的。

四. 符号说明

符号	含义
S	阈值
I	最大信息数
G	需要合成的样本
R	少数类样本对多数的占比
G	少数类样本合成的样本数
K	服务难度与提高满意度百分点的比例系数
C	三种品牌提升满意度百分点组合矩阵

五. 问题一模型的建立与求解

5.1 特征转换

特征转换是基于现有的特征，通过进行变换，得到新的特征，从而使得变换后的特征更加有利于数据分析或更加直观。在特征 B8 中，展示了客户体验者的出生年份。由于出生年份这一特征不够直观，本文对这一特征进行了转换，通过使用当前年份与该特征的取值作差，就得到了“年龄”这一新的特征。

5.2 异常值处理

5.2.1 基于箱线图的潜在异常值检测

当数据当中存在异常值时，尤其是存在着偏离较大的离群点时，会对数据分析和建模带来误差，因此，需要对异常值进行检测。常用的异常值检测方法包括 3σ 法则或 Z 分布方法，而这一类方法是以正态分布为假设前提的。由于本文的一部分数值型特征的分布并不符合正态分布，即数据分布不均匀。故本文选择使用对数据分布没有要求的箱线图，对数值型特性进行异常值检测。

使用箱线图对数据进行异常值检测的原理为：通过计算四分位数加减 1.5 倍四分位距，即是计算 Q1－1.5IQR 和 Q3+1.5IQR 的值，规定落在这一区间之外的数据为异常点。在箱线图中，可以看出变量数据的中位数、上四分位数、下四分位数、上下边缘和潜在异常点[3]。本文通过使用上四分位数代替数值大于 Q3+1.5IQR 的数据，使用下四分位数代替数值小于 Q1－1.5IQR 的数据，并绘制出了异常值处理前后的箱线图，如图 1 所示。

在图 5-1 中，中间部分的线表示中位数，箱子的上下边缘分别表示上四分位数和下四分位数，图中上方和下方的两条横线表示上下边缘，最上方和最下方的点即是潜在离群值。从图中可以看出，一部分特征在异常值处理之后箱线图变化明显，如：a1、a3、a5、B17 等，说明在进行异常值处理之后，这些特征的分布更加的集中，更有利于建模。此外，一些特征并没有检测出异常值，如：B2、B5、B7、B16，说明这些特征原本的数据质量较好。

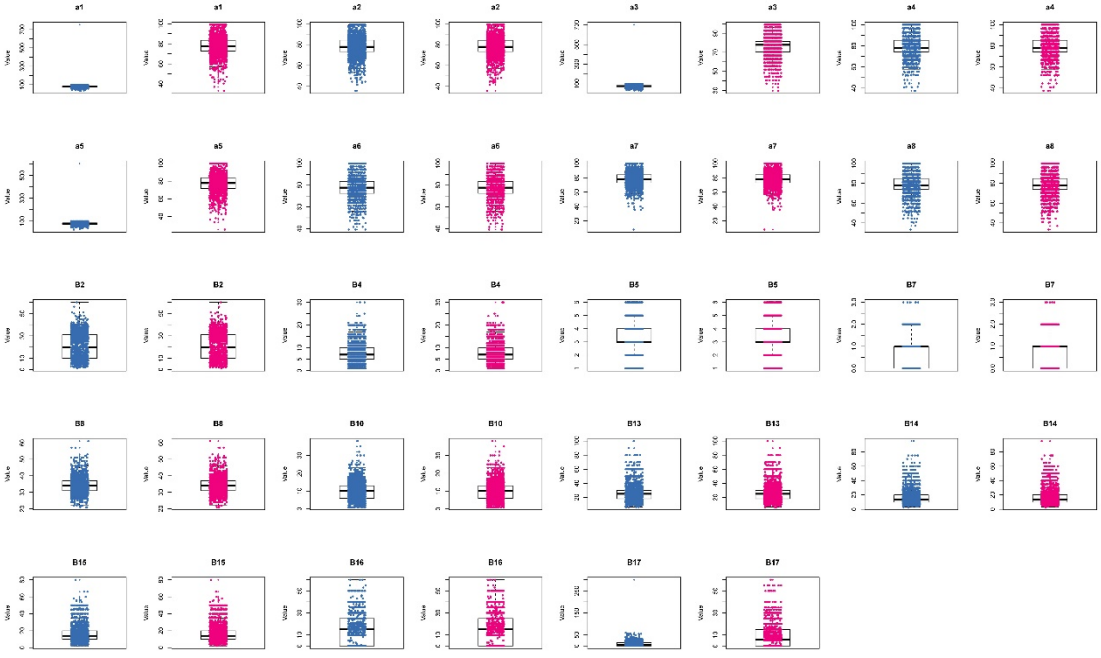

图 5-1 异常值检测箱线图

5.2.2 异常值识别与处理

通过对异常值进行分析我们发现，少量特征存在着个别数值较大的异常值，这些异常值与中位数的数值差距较大。在原数据中，我们经过观察、比较、分析之后发现，一部分异常值是人为导致的。其原因在于：这部分异常值为三位数，超出了满意度得分的满分 100 分这个最大值。同时，这样一些三位数的异常值的小数点后为 7 位，而正常的两位数的小数点后取值为 8 位。因此，我们判断出误差来源于小数点错位。对此，本文对这些异常值进行了更正处理，通过移动小数点，将三位数的异常值更正为两位数。

与此同时，对于 B16 和 B17 两个特征，由于是占家庭总收入的比例，故百分号前面的数字应小于 100，而在 B17 中，存在着 300 这一数字，显著大于阈值 100，故我们诊断其为异常值，并对其所在的行进行了删除处理。

5.3 缺失值处理

本文针对附录一中的目标客户体验数据，首先利用 Python 软件查找出不同特征下的缺失值分布情况，如图 5-2 所示。从图 5-2 可以看出，缺失值仅存在于 B7（孩子数量）这一特征中，且缺失值在数据集中的不同行的分布较为均匀，说明这些缺失值是随机缺失的。

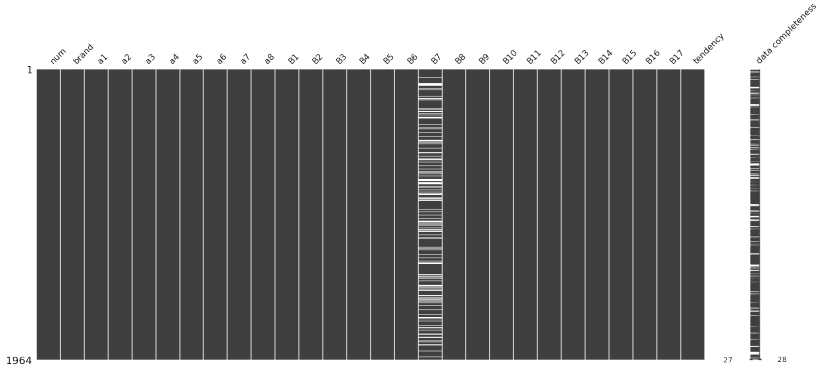

图 5-2 缺失值分布情况图

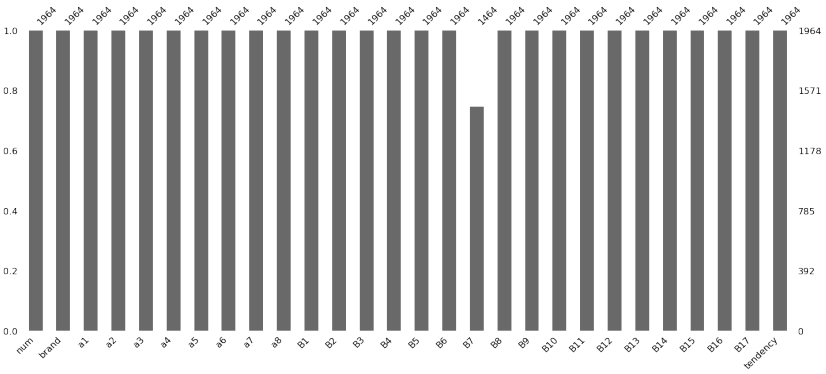

图 5-3 缺失值数量统计条形图

结合图 5-3，统计出 B7 所在列的缺失值个数为 500。本文进一步探究缺失值所在观测的内在联系，结合问卷的信息，可以明确目标用户的 B6（婚姻家庭情况）的取值，会限制 B7 的取值范围。

$$B7=\begin{cases}0, \text{when } B6 \text{ is } 1,2,3,4 \text{ or } 8\\ >0, \text{ when } B6 \text{ is } 5,6 \text{ or } 7\end{cases} \tag{5-1}$$

于是，本文统计出缺失值对应的 B6 列的取值情况，如表 5-1 所示。

表 5-1 缺失值对应 B6 取值情况

B6	1	2	3	4	5	7	8
数量	65	198	162	70	2	2	1

利用表 5-1 及 B7 的取值公式，本文将 B6 取值为 1,2,3,4 或 8，对应的 B7 列的缺失值填补为 0。并对于 B6 取值为 5 或 7 的情形，本文假定在 B6 真实的情况下，按照 B7 列非缺失的数据的中位数 1，作为在 B6 取值为 5 或 7 的缺失值取值。

5.4 描述性统计分析

为比较目标客户对于汽车品牌的满意度差异，本文绘制出三个品牌关于 8 个满意度变量的小提琴图，如图 5-4 所示。小提琴图刻画了数据的分布特征，这种图用来显示数据的分布和概率密度，可以看成是箱线图和密度图的结合。小提琴图的中间部分反映箱线图的信息，图的两侧反映出密度图的信息。其中，小提琴图中间的黑色粗条用来显示四分位数。黑色粗条中间的白点表示中位数，粗条的顶边和底边分别表示上四分位数和下四分位数，通过边的位置所对应的 y 轴的数值就可以看到四分位数的值。从小提琴图的外形可以看到任意位置的数据密度，实际上就是旋转了 90 度的密度图。小提琴图越宽的地方表示数据密度越大，可以展示出数据的多个峰值。

从图中可以看出，三个品牌对应的满意度得分的中位数都非常接近，而分布存在着差异。对于品牌类别 1，数据分布基本上只有一个明显的峰值，说明客户对于该种品牌的满意度较为一致。对于品牌类别 2，数据分布呈多峰形态，说明不同客户对该种品牌的满意度呈现多级分化。对于品牌类别 3，a1 电池技术性能满意度得分有两个分支，说明其满意度呈两极分布；a2-a8 大体上呈单峰分布，说明客户对于品牌 3 的 a2-a8 满意度较为统一。

为了量化不同汽车品牌满意度之间的具体差异，本文引入了 Wilcox 统计量和 Kruskal-Wallis 统计量[5]。其中，Wilcox 统计量用于比较两个总体分布之间的差异，Kruskal-Wallis 统计量用于比较 3 个总体分布之间的差异。在图 5-4 中，展示出了统计量的具体数值以及相应的 P 值，由 P 值可以得出，仅有特征 a5（动力性表现满意度得分）的 P 值小于显著性水平 0.1，表明客户对于不同品牌汽车的动力性能的满意度存在着较为明显的差异，而对于汽车其它性能的满意度差异不太明显。

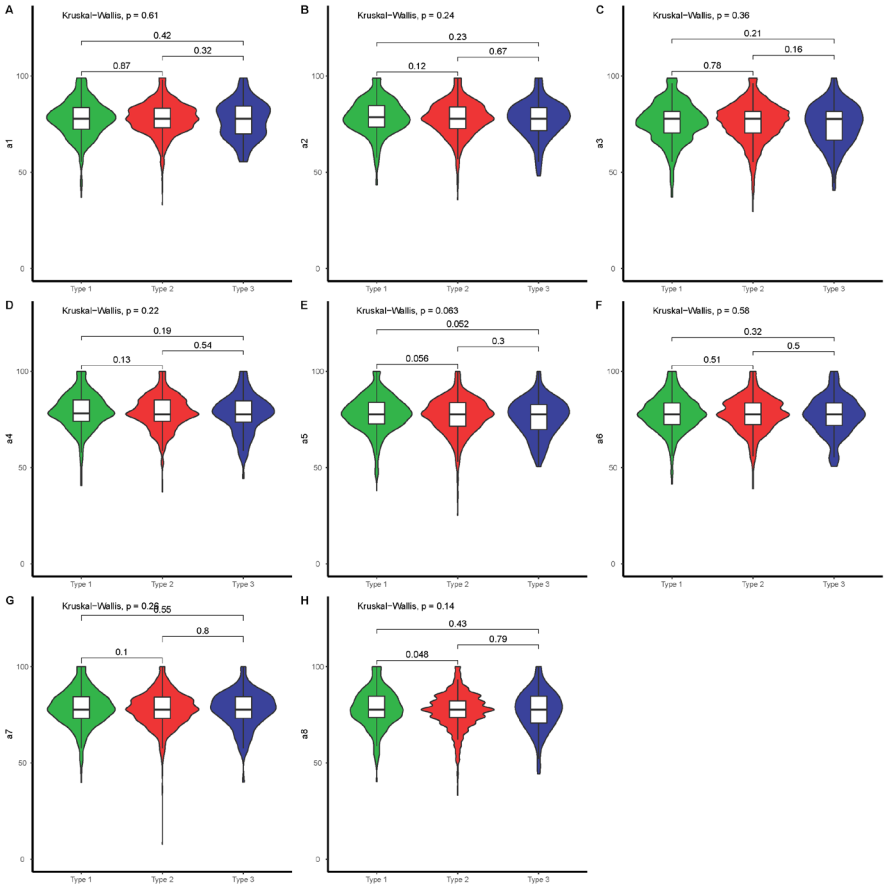

图 5-4 不同品牌汽车满意度小提琴图

接下来，本文进行了多变量的描述性统计分析，绘制热力图，如图 5-5 所示。探究了不同变量之间的相关性，进而探究自变量之间的多重共线性。热力图，又名相关系数图，根据热力图中不同方块颜色对应的相关系数的大小，可以判断出变量之间相关性的大小[6]。两个变量之间相关系数的计算公式为：

$$\rho_{x_1x_2}=\frac{Cov(X_1,X_2)}{\sqrt{DX_1,DX_2}}=\frac{EX_1X_2-EX_1*EX_2}{\sqrt{DX_1*DX_2}} \tag{5-2}$$

公式中，ρ 表示相关系数，Cov 表示协方差，E 表示数学期望，即均值。

在热力图中，右侧的刻度展示了不同相关系数对应的颜色深浅。从图中可以看出，a1-a8 这些变量所在的区域为蓝色，说明这些变量的相关程度较高。除此之外，（B10、B8）、（B13、B14、B15）之间也有着较强的相关性。对于相关性强的变量，往往存在着多重共线性。在进行特征工程时可以考虑剔除部分强相关的变量，以免导致因多重共线性造成的过拟合。

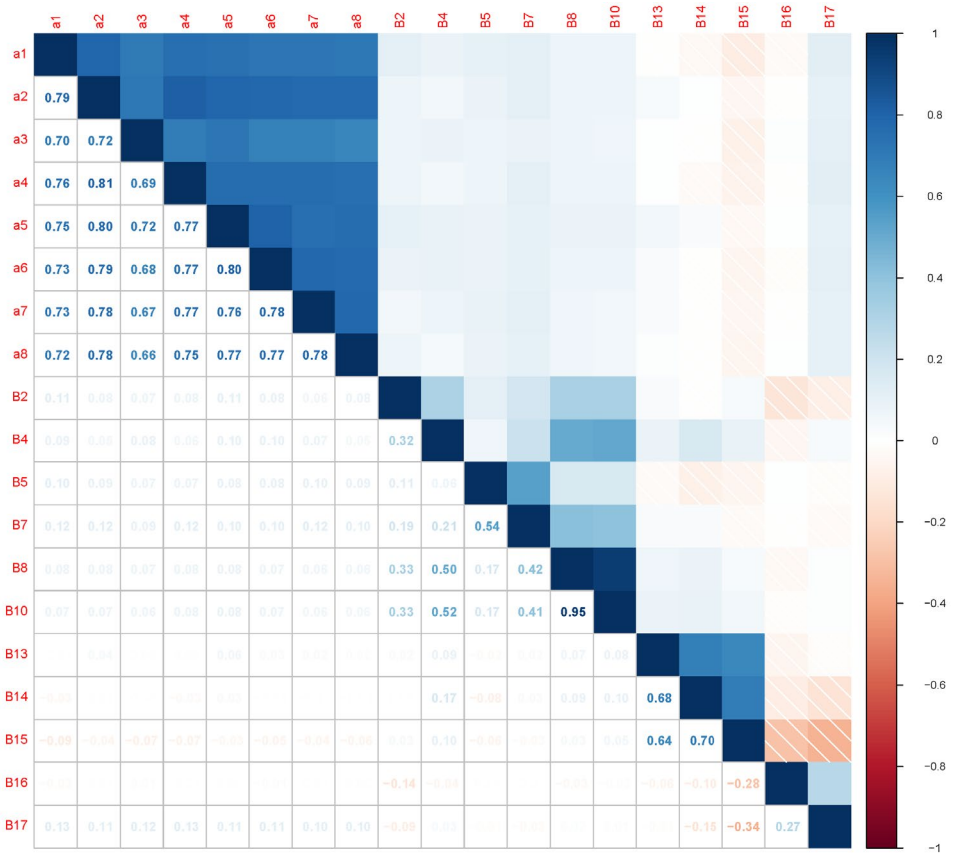

图 5-5 数值型变量相关性热力图

考虑到客户的居住区域对其购买行为往往会有一定影响，故本文选择对”居住区域”这一分类特征进行描述性统计分析。绘制树形图，如图 5-6 所示。树形图的面积代表了所对应类别的数量，常用于比较各类别在数量上的差异。从图中可以看出，居住在不同区域的目标客户数量差别明显。其中，居住在市中心的客户数量最多，有 1010 个，其次是居住在非市中心城区的客户数量，有 770 个。这从一定程度上表明电动汽车的目标客户主要集中在城市区域，而居住在农村和乡镇的目标客户数量极少。

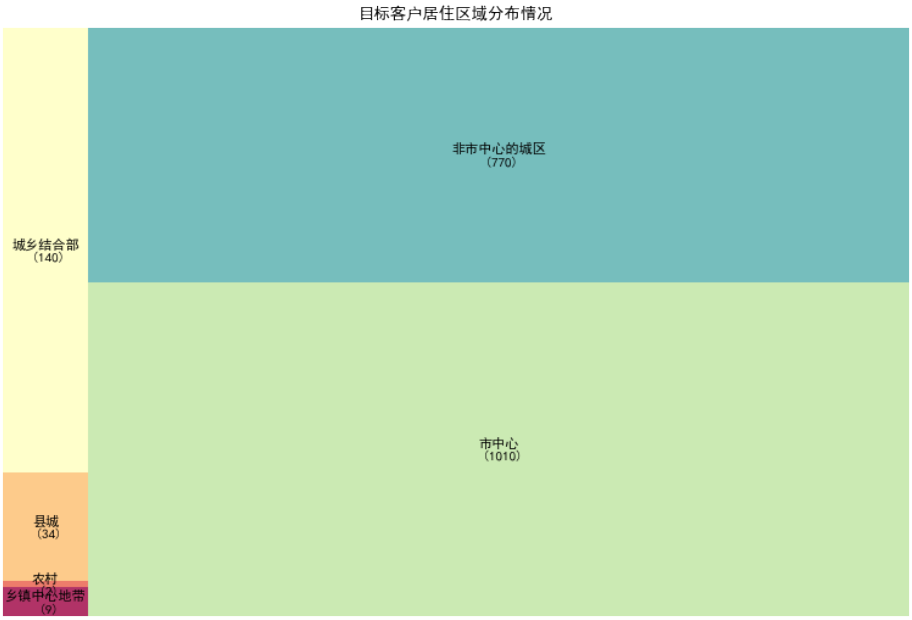

图 5-6 客户居住区域分布树形图

六. 问题二模型的建立与求解

6.1 数据归一化

为消除数据量纲的影响，需要对数值型变量进行最大最小归一化，将数据转化为 0 至 1 之间的数值。由于分类型变量在 One-Hot 编码后的取值只有 0 和 1，因此分类型变量无需进行归一化。最大最小归一化的计算公式为：

$$x^* = \frac{x - \min}{\max - \min} \tag{6-1}$$

（公式中 x^*表示标准化后的数据值，min 表示原数据中的最小值，max 表示原数据中的最大值）

6.2 ADASYN 采样

在附录 1 的所有数据样本中，有购买意愿的顾客数为 99，占比 5%，数据极度不平衡。如果直接用不平衡的样本筛选特征，会导致特征无法很好地预测少数类的样本；如果用不平衡的样本建立模型，会导致极高的正确率和极低的泛化能力。因此，本文对不同品牌的客户数据分别进行了采样处理。考虑到标签为“1”的样本数量较少，本文使用 ADASYN 上采样的方法，通过扩大少数类样本的数量，对训练集进行采样处理。

ADASYN 是一种自适应综合过采样的方法[7]，这种方法是根据样本的贡献来决定生成多少个合成数据。具体的实现过程如下：

1.计算不平衡度。记少数类样本为 ms，多数类样本为 ml，则不平衡度为

$$d = \frac{ms}{ml} \tag{6-2}$$

2.计算需要合成的样本数量 G

$$G = (ml - ms) * b, b \in [0,1] \tag{6-3}$$

3.对每个属于少数类的样本使用欧氏距离计算 k 个近邻，然后计算每个少数类样本周围多数类的情况，计算公式为

$$\hat{r}_i = \frac{r_i}{\sum_{i=1}^{m} r_i} \tag{6-4}$$

4.对每个少数类样本计算合成样本的数目 g_i

$$g_i = \hat{r}_i * G \tag{6-5}$$

5.在每个待合成的少数类样本周围的 k 个近邻中选择 1 个少数类样本，计算合成样本

$$s_i = x_i + (x_{zi} - x_i) * \lambda \tag{6-6}$$

6.3 分类型特征筛选

6.3.1 基于卡方检验的分类型特征初筛

经典的卡方检验是用来检验定性自变量与定性因变量之间的相关性。在卡方检验中，卡方值用来描述两个事件的独立性或者描述实际观察值与期望值之间的偏离程度。卡方值越大，则表明两个分类变量之间的独立性越弱，即相关性越强。卡方统计量的计算公式为：

$$\chi^2 = \sum \frac{(A-E)^2}{E} \tag{6-7}$$

在公式中，A 表示实际频数，E 表示期望频数。本文分别对三种汽车品牌的 7 个分类变量与“购买意愿”这一目标变量分别进行卡方检验，并计算出卡方检验结果的得分和 P 值。通过设置得分阈值为 S，得到了大于阈值的 N1 个变量。值得注意的是，由于本文的特征筛选由多个步骤组成，因此此处通过人为设定阈值并不会对最终的特征集造成影响，因为如果存在少量不合理的特征，在后续的特征筛选中也会被过滤掉。三种汽车品牌设定的得分阈值和初步筛选得到的分类变量如表 6-1 所示。

表 6-1 分类特征一次筛选特征集

品牌类型	卡方检验得分阈值	分类特征一次筛选后的特征集
1	1	B1、B3、B6、B12
2	10	B3、B9、B11、B12
3	1	B3、B6、B11、B12

6.3.2 基于最大信息系数法的分类型特征二次筛选

在前面筛选得到 4 个分类型变量的基础上，使用最大信息系数法对分类变量进行二次筛选[8]。在许多时候，变量之间的关系都是非线性的，而最大信息系数法适用于捕捉出变量之间的非线性关系。最大信息系数，又称为 MIC，是把互信息取值转化为一种度量方式。最大信息系数通常适用于大样本的数据集，其计算公式为：

$$I(X;Y) = \sum_{x \in y} \sum_{y \in x} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{6-8}$$

本文通过设置 K=2，最终筛选得到了 2 个最优的分类变量。三种汽车品牌二次筛选得到的特征集如表 6-2 所示。

表 6-2 分类特征二次筛选特征集

品牌类型	分类特征二次筛选后的特征集
1	B3、B12
2	B3、B12
3	B11、12

6.4 数值型特征筛选

6.4.1 基于方差过滤法的数值型特征初筛

在数值型特征筛选的第一阶段，本文使用了过滤式中的方差选择法。这种方法是通过分别计算各个特征的方差，在此基础上选出方差大于阈值的特征。对于方差小的特征，由于其取值与均值接近，在建模过程中对模型的贡献程度往往较低，故需要去掉这一类特征；而对于方差较大的特征，其分布往往更为分散，这类特征对于模型有着更好的扰动性，往往会有更好的分类预测效果。方差的计算公式为：

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{6-9}$$

公式中，σ^2 是方差，n 是样本数量，x_i 是特征 x 的第 i 个值，$\bar{x}$ 是特征 x 的期望。在 19 个原始数值型特征的基础上，通过设置方差阈值为 C，得到了大于方差的 13 个候选指标。三种汽车品牌对应的方差阈值和初步筛选得到的数值变量如表 6-3 所示。

表 6-3 数值特征一次筛选特征集

品牌类型	方差阈值	数值特征一次筛选后的特征集
1	0.14	a1, a2, a3, a4, a6, a8, B2, B4, B5, B7, B13, B16, B17
2	0.13	a1, a2, a3, a4, a6, a8, B2, B4, B5, B7, B10, B16, B17
3	0.15	a2, a3, a4, a6, a8, B2, B4, B5, B7, B13, B15, B16, B17

6.4.2 基于随机森林的数值型特征二次筛选

在特征选择的第二阶段，本文使用嵌入式的方法。在前面筛选出的 13 个候选指标中，通过随机森林度量各特征的重要性程度。随机森林是以决策树作为的基分类器，CART 决策树是一种典型的二叉树。在构建 CART 决策树过程中，以 Gini 指数最小的属性来作为最优划分属性。Gini 指数也叫节点不纯度减少平均值[9]，特征的 Gini 指数越小，说明该特征就越重要。通过决策树分类器对各个特征区间的划分，可以得到各个特征的重要性程度。这里以品牌 1 为例，将特征按重要性程度从高到低进行排序后，绘制出特征重要性程度图，如 6-1 所示。

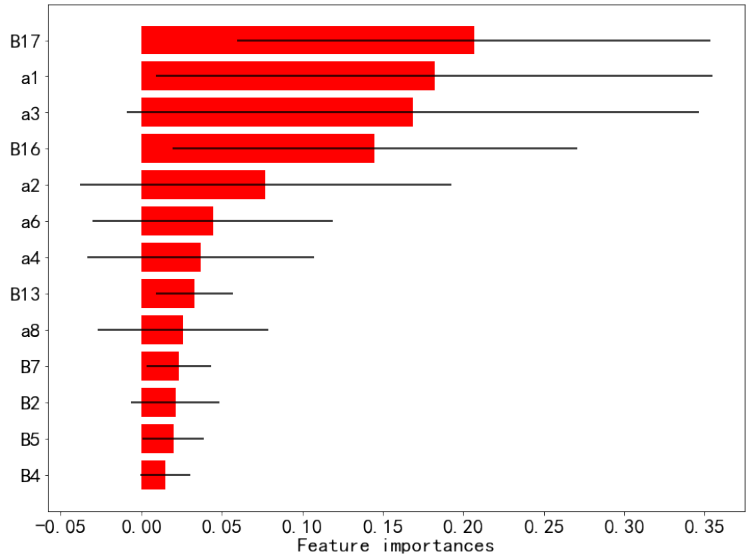

图 6-1 特征重要性程度图

本文通过删除重要性程度排序最靠后的 5 个特征，得到了三种品牌的数值特征二次筛选后的特征集，如表 6-4 所示。

表 6-4 数值特征二次筛选特征集

品牌类型	数值特征二次筛选后的特征集
1	a1, a2, a3, a4, a6, B13, B16, B17
2	a1, a2, a3, a4, a6, a8, B16, B17
3	a2, a3, a4, a6, a8, B4, B13, B16

6.4.3 基于 RFE 递归特征消除的数值型特征三次筛选

由于前面使用随机森林筛选出的特征结果并不是绝对稳定的，该方法由于随机种子设定的不同，筛选出的特征也不全相同。为了使最终建模的特征更加稳定客观，本文使用递归特征消除的方法对数值型特征进行第三次筛选。在递归特征消除法中，需要使用一个基模型来进行多轮训练，本文选取的基模型是梯度提升树 GradientBoost[10]。在基模型训练之后，会得到每个特征的权值系数，然后将权值系数绝对值最小的特征进行剔除，如此不断循环递归，直至剩余的特征数量达到所需的特征数量。本文最终保留了 4 个数值型特征。

表 6-5 数值特征三次筛选特征集

品牌类型	数值特征三次筛选后的特征集
1	a1, a3, B16, B17
2	a1, a4, B16, B17
3	a2, a3, a6, B16

经过分类变量的筛选和数值变量的筛选，本文最终分别为三种品牌确定出 6 个特征，即电动汽车购买的影响因素，用于后面的建模，如表 6-6 所示。

表 6-6 最终筛选得到的影响因素

品牌类型	序号	影响因素汇总
1	1	电池技术性能满意度得分 a1
	2	经济性满意度得分 a3
	3	房贷占家庭总收入比例 B16
	4	车贷占家庭总收入比例 B17
	5	居住区域 B3
	6	职位类型 B12
2	1	电池技术性能满意度得分 a1
	2	安全性满意度得分 a4
	3	房贷占家庭总收入比例 B16
	4	车贷占家庭总收入比例 B17
	5	居住区域 B3
	6	职位类型 B12
3	1	舒适性满意度得分 a2
	2	经济性满意度得分 a3
	3	驾驶操控性表现满意度得分 a6
	4	房贷占家庭总收入比例 B16
	5	所在单位性质 B11
	6	职位类型 B12

七. 问题三模型的建立与求解

7.1 算法介绍

7.1.1 XGboost 算法框架模型

XGboost 由 GBDT、RGF 等算法改进而来[11]，是一种基于 Boosting 的集成学习算法。XGboost 的基学习器可以是决策树，也可以是线性模型。对于有少量缺失值的数据集，该模型具有较好的容错能力，可以通过稀疏感知算法自动学习出决策树的分裂方向。在决策树的分裂过程中，使用贪心算法的近似算法寻找最有可能的分裂点。由于采用分布式计算，需要遍历所有的数据集，XGboost 会更加地消耗计算机内存。

XGboost 首先根据特征对数据进行排序，将排序后的特征储存到块结构中，并且通过稀疏矩阵储存格式来减少计算量。在特征分块并排序的过程中，通过顺序访问排序后的特征值，以便于进行切分点的查找。该模型对特征使用并行化处理方式，选择信息增益最大的特征作为分裂方向，并且多个特征的增益计算同时进行。XGboost 算法的具体原理如下：

①构造损失函数与目标函数

通过构造损失函数，来减少模型的偏差和方差。减小偏差即是减少模型预测结果和真实值之间的误差。减小方差可以看成是防止过拟合，一般通过在模型中引入正则项和降低模型复杂度来实现。目标函数由损失函数和正则项构成，计算公式如下：

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{i=1}^{t} \Omega(f_i) \tag{7-1}$$

由于 Boosting 遵从前向分布加法，每一步的预测值都由前一步的预测值所确定。所以通过叠加，我们可以得到目标函数的最终表达式：

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_i) + cons\tan t \tag{7-2}$$

②目标函数的泰勒展开近似

因为 XGboost 的目标函数过于复杂而难以直接求解，这里使用泰勒多项式来逼近目标函数。泰勒公式是将一个在 $x = x_0$ 处具有 n 阶导数的函数 $f(x)$利用关于 $(x - x_0)$ 的 n 次多项式来逼近函数的方法。根据泰勒公式，把函数 $f(x + \Delta x)$ 在点 x 处进行泰勒公式的二阶展开，就可以得到下面的等式：

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2 \tag{7-3}$$

因此，对于 XGboost 的目标函数，将前一个公式的 $f(x)$ 对应于损失函数 $l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$，将 x 对应于前 t-1 棵树的预测值 $\hat{y}_i^{(t-1)}$，将 Δx 对应于正在训练的第 t 棵树，就可以得到目标函数的近似函数。在去掉对函数优化无影响的常数项之后，就可以将近似目标函数写为：

$$Obj^{(t)} \simeq \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{7-4}$$

③决策树的生成

对于决策树的复杂度，由叶子数 T 和叶子节点的权重 w 来确定。通过引入 L_2 范式作为正则项，可以得到复杂度的计算公式：

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{7-5}$$

接下来，通过对树的结构进行打分，使用贪心算法寻找树的分裂收益，得到树分裂后的目标函数为：

$$Obj = -\frac{1}{2}[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}]+2\gamma \quad (7\text{-}6)$$

XGboost 算法的优点在于：第一，通过对损失函数进行二阶展开，相较于只用到一阶展开的 GBDT 算法，具有更高的准确度。第二，该模型通过引入正则项来降低模型的复杂度，可以有效的防止模型过拟合。第三；由于 XGBoost 通过稀疏感知算法自动学习出树的分裂方向，在预处理阶段，对于少量的缺失值，可以不予处理。XGboost 的缺点在于：由于在树的节点分裂过程中需要遍历数据集，大大地消耗了计算机内存。

7.1.2 LightGBM 算法框架模型

LightGBM 的主要思想是通过决策树的迭代训练以得到最优模型[12]。作为一个实现 GBDT 算法的框架，LightGBM 具有训练速度快，消耗内存少的特点。这实际上是一种基于 Histogram 的决策树算法，通过单边梯度采样，只保留梯度高的数据。并且利用保留的高梯度数据计算出信息增益。Histogram 也被称作直方图算法，通过对连续型的数据进行分箱处理，大大的减少了数据内存。一方面，数据离散化为对结果造成误差；另一方面，通过这种粗糙分割的方式，使得学习器在训练的过程中，防止过拟合。同时，LightGBM 在 Leaf-wise 之上增加了一个最大深度限制，也在一定程度上防止了过拟合。

LightGBM 利用 Histogram 做差进行优化加速，先计算出直方图中较小的叶子节点，然后通过直方图做差来获得较大的叶子节点。在单边梯度采样过程中，LightGBM 引入了 GOSS 采样算法，首先将要进行分裂的特征按照绝对值从大到小进行排序。接下来，分别选取绝对值较大的数据和绝对值较小的部分数据。由于在训练过程中，梯度较小的数据往往会训练不足，故这里对绝对值较小的数据给予更高的权重。最后，使用选出的这些数据计算出信息增益。

该模型使用互斥特征捆绑算法来达到降维的目的。对于高维且稀疏的数据，将非零的互斥特征进行捆绑。这样，就降低了直方图构造过程中的时间复杂度。LightGBM 提出了一种无图的排序策略，通过将特征按照非零值的个数进行排序，来确定将哪些特征捆绑在一起。

LightGBM 的一大特点在于可以直接支持类别特征。对于一般的分类模型，类别特征都需要进行独特编码处理，然而，One-Hot 编码会给数据造成稀疏性，并且在类别数较多时，无法保证数据的平衡。实际上，LightGBM 通过 many-vs-many 的切分方式将类别特征分为两个子集，从而实现类别特征的最优切分。即是先把直方图按照每个类别对应的标签均值进行排序，然后按照排序的结果依次枚举出最优的分割点。

该模型支持高效并行。在特征并行上，分别保存全部数据并基于此前得到最佳划分条件执行接下来的训练。在数据并行上，使用分散规约的方法，把直方图合并后的数据分摊至不同的学习器，并进行做差，从而减少了计算量。在投票并行上，只合并部分特征的直方图，通过找出重要程度高的特征，基于投票法来确定出最优的分割点，以起到加速的效果。

LightGBM 的优点在于：第一，在模型的训练过程中，用遍历直方图来替代遍历样本，降低了时间复杂度；第二，使用单边梯度算法过滤掉小梯度的样本，减少了计算量；第三，采用互斥特征捆绑算法来减少特征的数量，以降低内存消耗。LightGBM 的缺点在于：第一，通过多次迭代对样本的权重进行调整，虽降低了误差和偏差，但容易受到数据噪声的影响；第二，在模型寻找最优解的过程中，并没有考虑到全部特征。

7.1.3 CatBoost 算法模型

CatBoost 是以对称决策树作为基学习器[13]，通过嵌入一种自动将类别特征处理为数值型特征的算法，对于类别特征有着准确高效地处理效果。对称树是一种平衡树，在树的同一层使用相同的分割准则，树的每个叶子节点的索引可被编码为长度与树深度相等的二进制向量，以此来达到快速评分的效果。

CatBoost 使用改进 Greedy TS 的方式实现目标变量的统计。该模型在将标签平均值作为节点分裂标准的基础上，通过添加先验分布项，以减小数据噪声和出现较少的类别特征。计算公式为：

$$\hat{x}_k^i = \frac{\sum_{j=1}^{p-1}[x_{\sigma_{j,k}} = x_{\sigma_{p,k}}]Y_{\sigma_j} + \alpha * p}{\sum_{j=1}^{p-1}[x_{\sigma_{j,k}} = x_{\sigma_{p,k}}] + \alpha} \tag{7-7}$$

公式中，p 是添加的先验项，α 通常是大于 0 的权重系数。同时，CatBoost 利用了一种计算叶子节点值的方法，以避免数据集排列过程中直接计算会出现的过拟合问题。CatBoost 可以将数值型特征和类别型特征进行组合，这种组合方式是通过将决策树中选定的所有分割点视为具有两个值的类别特征，对其像类别特征一样进行组合考虑。

对于 CatBoost 处理类别特征的方式，首先计算出某个类别出现的频率，对其加上超参数后，就生成了新的数值型特征。接下来，使用数据的不同排列，通过随机的方式来选择生成树的排列。之间，将不同的类别特征进行组合，在选择第一个节点时，先只选择一个特征；当生成至第二个节点时，则考虑前一个特征和其它任意一个类别特征的组合，并使用贪心算法选择其中的最优特征组合。

在传统的梯度提升算法中，往往难以避免有偏梯度估计引起的过拟合问题。CatBoost 对其进行了改进，在构建树的第一阶段，采用的是梯度步长的无偏估计，第二阶段则和传统 GBDT 相同。

在模型的训练过程中，CatBoost 通过以下方式来实现 GPU 快速训练。第一，以 oblivious 决策树作为基模型，将特征进行了离散化保存，这种直方图计算方法不依赖于原子操作，实现起来更加地高效。第二，对类别特征使用完美哈希进行按位压缩。第三，采用分布式的学习方式，对多个数据集并排计算，可实现多个 GPU 加速。

CatBoost 的优点在于：第一，需要优化调整的参数较之其它模型更少，具有噪声鲁棒性；第二，通过特征组合，对于类别特征和数值特征有着良好的处理效果；第三，支持自定义损失函数，模型的灵活性较高。CatBoost 的缺点在于：第一，在处理类别特征时，较之其它模型会消耗更多内存；第二，模型的预测结果会受到随机数设定值的影响。

7.1.4 随机森林算法模型

随机森林是以决策树为基学习器的一种集成学习方法[14]，与 Bagging 不同的是，其引入了随机属性选择。属性扰动使得个体学习器之间差异度的增加而进一步提升，从而提高模型的泛化能力。随机森林模型作为若干棵决策树的组合，弥补了单棵决策树分类能力弱的不足，可以将输入和输出之间的复杂关系平均在很多简单的决策树上面，从而提高了分类的准确率。

由于随机森林是单棵决策树的集成，故需要先构建单棵决策树。具体步骤可分为构建树，剪枝，集成树三个阶段。

①建树

在构建决策树过程中，通过设置一个期望误差减少值，作为树节点的分裂属性，设定当标准差小于 5%时，树将终止分裂。标准差减少值计算如下：

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} * sd(T_i) \quad (7\text{-}8)$$

（其中T_i表示节点分类分裂的结果数据集）

②选择分裂属性

在二分类问题中，随机森林集成了多棵 CART 二叉树。利用 Gini 指数作为 CART 算法的分区准则，在决策树的每个内部结点上选择属性，其计算公式为：

$$Gini(S) = 1 - \sum_{i=0}^{c-1} p_i^2 \quad (7\text{-}9)$$

公式中，S 表示数据集，C_i 是类，S_i 是属于类的样本数，$p_i = s_i / S$ 是类 C_i 在集中的相对频度。通过决策树分类器对各个特征区间的划分，可以得到各个特征的重要性程度。

②剪枝

在构建好决策树之后，由于部分树的分支只能捕捉到极少的样本，故需要将这些分支减去。CART 中使用的是最小成本复杂度修剪技术。根据误分率对应的节点数量，在误分率满足的条件下，对分枝对应的特征进行提纯。

③集成树

对多棵决策树进行集成，可以有效的避免单棵决策树的不稳定性。通常情况下，集成的决策树的数量过少，模型可能无法稳定收敛；集成的决策树数量过多，则消耗计算机内存。一般设置集成 1000 棵决策树，可以在减少计算资源消耗的前提下，使得模型收敛。

随机森林的优点在于：第一，适用于高维数据；第二，可以判断出特征之间的相互影响和重要性程度；第三，训练速度较快，不易造成过拟合。随机森林的缺点在于：对于噪声较大的数据，不具备较好的容错能力。

7.2 模型建立

7.2.1 One-Hot 编码

由于一些分类模型不能直接支持类别变量，因此在这些模型中，需要对分类变量进行 One-Hot 编码。One-Hot 编码又称为独热编码，是将类别变量转换为机器学习算法易于利用的一种形式的过程。这针对于分类型的数据，One-Hot 编码主要是采用 N 位状态寄存器来对 N 个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。使用 one-hot 编码将离散特征的取值扩展到欧式空间，离散特征的某个取值即对应欧式空间的某个点。在进行编码之后，原特征的 n 个类别会转化为 n-1 个特征，从而避免了数据的共线性。

7.2.2 交叉验证

为了保证模型的训练效果和准确性，本文使用 K 折交叉验证的方法。交叉验证是用来验证模型性能的一种方法，基本思想是将一部分数据划分为训练集，另一部分数据划分为验证集。通过将数据分为 K 组，每组数据分别做一次验证集，总共会得到 K 个模型。对这 K 个模型的准确性取平均值就得到了最终模型的评价结果。K 折交叉验证的优点在于对数据集的划分更加的细致，避免了纯随机划分样本带来的误差。

7.2.3 贝叶斯优化调参

如果直接进行模型训练，往往难以找到准确的模型参数，因此，本文使用贝叶斯调参的方法，对模型的参数进行了优化。贝叶斯调参是一种自动化机器学习方法[15]，本文以 $F1$ 值作为优化目标，使用贝叶斯优化的方法分别对 XGboost、LightGBM、Catboost 和随机森林这四个模型进行调参。贝叶斯调参的优势在于可以自动搜索出模型的超参

数，适合于参数较多的模型的搜索。相较于网格搜索调参，贝叶斯调参大大减少了内存的消耗，并且结合了先验概率，因而更加高效、准确。

贝叶斯优化的基本思想是通过多次迭代来逐步逼近最优参数，使用贝叶斯定理来估计目标函数的后验分布，进而根据分布选择下一个超参数组合。使用高斯过程来拟合优化目标函数，再使用贝叶斯优化，来搜索出最优的参数值。贝叶斯优化通过不断地更新概率模型，使得后验分布逼近于真实分布，从而寻找到使全局提升最大的参数。

由于不同模型的特点不同，所用到的参数也不尽相同。受本文篇幅所限，下面以品牌 1 中效果表现最优的 LightGBM 模型作为调参结果的解释，调参结果如表 7-1 所示。

表 7-1 LightGBM 参数说明及调参结果

参数名称	参数简称	参数含义	调参结果
Target	目标	目标函数值	0.875
Max depth	最大深度	指树的最大深度。树的最大深度过小会导致欠拟合，即模型训练不充分；树的深度过大可能会导致过分学习，即过拟合	14
n_estimators	树的估计量	指模型中树的个数。理论上树的个数越多则越精确，但需要考虑计算成本	57
min_samples_split	最小样本分割	一个节点必须最少包含的训练样本数量，使得节点被分割	10.608
max_features	最大特征数	在对树进行划分时，只考虑部分特征，以控制树的生成时间	0.683

7.3 模型评估

在实际的汽车购买场景中，我们更关心客户是否会购买汽车，即是取值为“1”的样本被准确预测的比例，因此本文使用 AUC 值和 *F*1 值对模型的预测效果进行评估。AUC 值被定义为 ROC 曲线下的面积，用于反映模型整体的预测效果。*F*1 值综合了查准率和查全率，相当于是二者的调和平均值,用于反映购买汽车的客户的预测效果。其中，查准率表示在预测出的购买汽车的客户中实际购买汽车的客户占比；查全率表示在实际购买汽车的客户中，成功预测出的购买汽车的客户数量的占比。F1 值的计算公式为：

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{7-14}$$

公式中，precision 表示查准率，recall 表示查全率。三种汽车品牌的四种模型的评估结果如下表所示。

表 7-2 品牌 1 模型评估结果

	随机森林	LightGBM	Catboost	XGBoost
AUC 值	0.9856	0.9856	0.9784	0.9712
查准率	0.8571	0.7778	0.7	0.6667
查全率	0.8571	1.0	1.0	0.8571
*F*1 值	0.8571	0.8750	0.8235	0.75

表 7-3 品牌 2 模型评估结果

	随机森林	LightGBM	Catboost	XGBoost
AUC 值	0.9769	0.9789	0.9733	0.9859
查准率	0.7192	0.8367	0.6752	0.7812
查全率	0.81	0.8367	0.8633	1.0
*F*1 值	0.7619	0.8367	0.7578	0.8772

表 7-4 品牌 3 模型评估结果

	随机森林	LightGBM	Catboost	XGBoost
AUC 值	0.94235	0.97721	0.89411	0.96294
查准率	0.8231	0.7864	0.7	0.7778
查全率	0.6667	0.8341	0.7	0.8347
*F*1 值	0.7367	0.8095	0.7	0.8052

7.4 模型预测

在建立完模型的基础上，分别选取三种品牌中表现最优的模型，对附录 3 中的客户数据进行预测。本文分别预测出客户是否会购买以及购买的可能性。预测结果如表 7-5 所示。

表 7-5 客户购买行为和可能性预测结果

客户编号	品牌编号	预测的模型	是否会购买 (0 表示买，1 表示不买)	购买的可能性
1			1	0.6776
2			0	0.4154
3	1	LightGBM	0	0.0019
4			0	0.0015
5			0	0.0270
6			1	0.92271
7			1	0.69083
8	2	XGBoost	0	0.29834
9			0	0.03425
10			1	0.54477
11			1	0.54477
12			1	0.80810
13	3	LightGBM	0	0.31870
14			0	0.06114
15			0	0.18126

八. 问题四模型的建立与求解

8.1 模型建立

本文在对于不同方面的整体满意度提升与目标用户购买意愿分析上，首先假定提升的整体满意度百分点为整数，结合第三问建立的不同品牌电动汽车的客户挖掘模型，提出提升满意度优化模型。

8.1.1 提升满意度优化模型

由已知信息，可知加大服务力度，提高 a1-a8 五个百分点的满意度在短的时间内是有可能的，故 8 个整体满意度的提升百分点取值，限制在 0 到 5 之间，从而所有不同类别的整体满意度的提升百分点组合是有限的。结合不同品牌的输入样本矩阵 $X=(x_1,x_2,x_3)$，其中 x_i 是第 i 个品牌中，挑选的目标客户信息向量。将其作为预测样本，代入到 LightGBM 模型中，得到不同提升组合下的客户购买意愿标签，提取出客户愿意购买电动汽车（lable=1）的数据，由此建立提升满意度优化模型。

首先，由于服务难度与提高的满意度百分点是成正比的，所以提升满意度优化模型中的目标函数是要使得各项整体满意度的服务难度最小化。以此为基础，得到的约束条件与目标函数为[16]

$$\min f(C)=\sum_{i=1}^{8} kc_{ij} \quad j=1,2,3$$

$$s.t.\begin{cases} x_{ij}+c_{ij}\le 100 & i=1,2,3\ \ j=1,2,\cdots,8 \\ 0\le c_{ij}\le 5 & i=1,2,3\ \ j=1,2,\cdots,8 \\ c_{ij}\in N^{*} & i=1,2,3\ \ j=1,2,\cdots,8 \\ g_i(a_i\cdot x_i+b_i)=1 & i=1,2,3 \\ sign(c_{ij})=sign(a_{ji}x_{ij}) & i=1,2,3\ \ j=1,2,\cdots,8 \end{cases} \tag{8-1}$$

其中，$g_i(x)$ 表示第三问根据第 i 种品牌建立的客户挖掘模型；b_i 表示第 i 种品牌的提升满意度调整向量，对应的调整矩阵 $B=(b_1,b_2,b_3)$，同时 $B=\begin{bmatrix} C \\ O \end{bmatrix}$；$a_i$ 表示第 i 种品牌的特征提取向量，对应的特征提取矩阵 $A=(a_1,a_2,a_3)^T$，同时

$$A=\begin{bmatrix} 1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&1 \\ 1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0 \end{bmatrix}$$

由于 $g_i(x)$=1 的约束，该模型不能通过现有的规划算法求解，本文在提升满意度优化模型的基础上，结合网格搜索法，来遍历在不同提升满意度组合的情形下，满足约束条件为 $g_i(x)$=1 的所有组合。

8.1.2 网格搜索法

网格搜索法是指定参数值的一种穷举搜索方法，通过将估计函数的参数通过交叉验证的方法进行优化来得到最优的学习算法[17]。也即，将各个参数可能的取值进行排列组合，列出所有可能的组合结果生成“网格”。然后将各组合用于所需要训练的机器学习算法中，并使用交叉验证对表现进行评估。

本文利用网格搜索法的调参思想，将各个整体满意度的提升百分点可能的取值进行排列组合，并合并附件三中选择的客户信息向量，结合提升满意度优化模型的约束条件，

得到更新后的待测样本，将其导入到 LightGBM 模型中，得到每一个“网格”（满意度提升百分点组合）的分类标签，并提取出目标客户愿意购买电动汽车的满意度提升百分点组合，如图 8-1 所示。

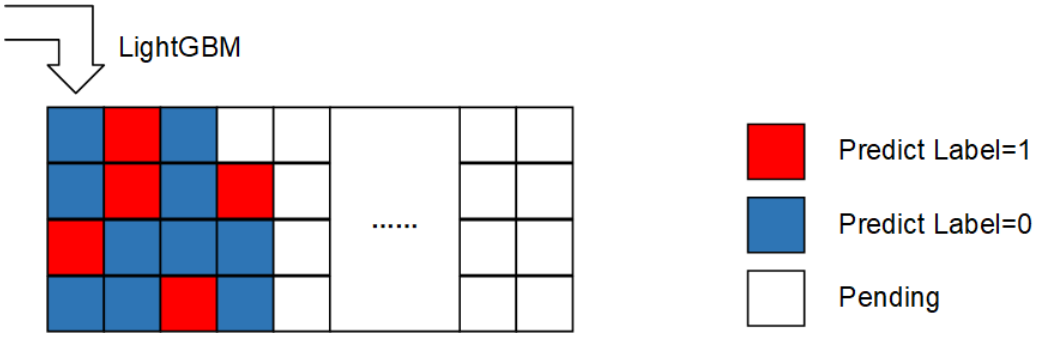

图 8-1 基于 LightGBM 模型的网格搜索示意图

结合图 8-1 以及提升满意度优化模型，以合资品牌为例，设计出上述规划模型的求解算法，以下是算法步骤：

Step1: 初始化 $f_0=0,\ E_0=\inf$；

Step2: 输入针对合资品牌的所有提升满意度组合的向量合并的矩阵 C，以及待测目标用户信息向量 x，其中 $C=\{c_0,c_1,c_2,\cdots,c_m\}$；

Step3: 递推公式：

$$f_i=f(c_i+x) \tag{8-2}$$

$$E_i=\begin{cases}\sum_{i=1}^{N}kc_i, & 当f_i=1,\\ \inf, & 当f_i=0.\end{cases} \tag{8-3}$$

$$\varphi_i=\begin{cases}c_i, & 当f_i=1,\\ \inf\cdot 1_{N1}, & 当f_i=0.\end{cases} \tag{8-4}$$

$$\delta_{i+1}=\min\{E_{i+1},E_i\} \tag{8-5}$$

其中，f 是基于合资品牌的客户挖掘模型；E_i 是第 i 种组合方式中，所有满意度提升的百分点所需要的难度；φ_i 是记录模型预测出客户愿意购买时，对应的提升满意度组合；δ_{i+1} 是最小难度。

Step4:输出 δ_m 与更新后的满意度提升百分点矩阵 Ψ，其中 $\Psi=\{\varphi_1,\varphi_2,\cdots,\varphi_m\}$。

8.1.3 服务提升标准评价体系

标准评价体系在企业战略经营管理中，又可以写成决策矩阵[18]，它是与决策方案与有关因素之间相互关系的矩阵表式，常根据决策矩阵做定量决策分析。

本文为了更好的从模型输出结果中，制定销售策略，并且考虑到各项服务项目的提升百分点均为整数，结合提升满意度优化模型输出的结果以及最优化提升满意度组合，建立服务提升标准评价体系，以便于销售策略的制定。

表 8-1 服务提升标准评价表

可能取值	0	1	2	3	4	5
服务提升等级	无服务	低	较低	中等	较高	高

根据表 8-1，将模型输出的结果转化成对应服务提升等级，能够提供更加具有可操作空间的销售策略。

8.2 提升满意度优化模型求解

本文在问题三建立的 LightGBM 目标客户挖掘模型对附件三的所有目标客户的购买意愿预测基础上，选定附件三中不同品牌下的没有购买意愿的目标客户（客户编号为 2,8,15），作为问题四的输入样本，并对选定的客户制定销售策略。这里绘制出 8 种服务项目下的 3 种不同品牌样本的柱形图，如图 8-2 所示。

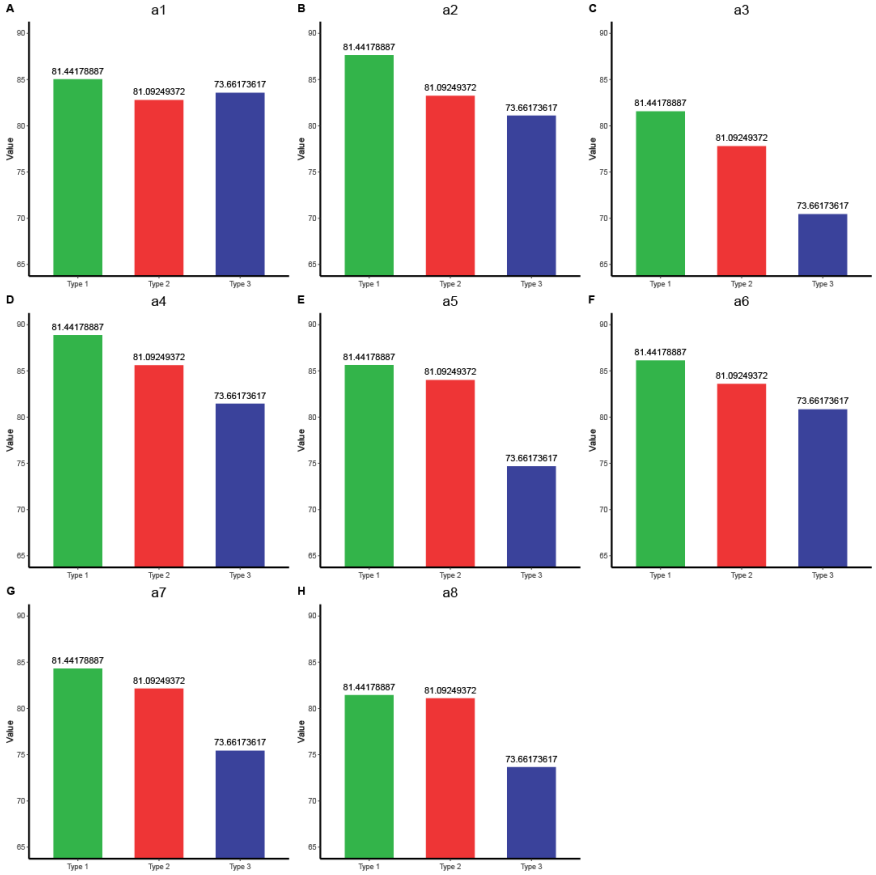

图 8-2 3 种品牌输入样本满意度柱形图

将不同提升满意度组合结合输入样本，作为新的待测样本，输入到 LightGBM 模型中，利用网格搜索法，最终得到使得客户购买意愿为 1 的所有满意度组合，最后利用 Python 软件，求得不同品牌的目标客户 8 项服务项目满意度，如表 8-2 所示,其中 c1~c8 代表对应分量下的服务项目整体满意度提升百分点。

表 8-2 最优满意度提升百分点组合

客户	c1	c2	c3	c4	c5	c6	c7	c8	label
2	2	0	0	0	0	0	0	0	1
5	0	0	1	1	0	0	0	0	1
8	0	0	2	0	0	0	0	0	1
	0	0	1	0	0	1	0	0	1

结合表 8-2 与服务提升标准评价体系，对三种品牌的目标客户制定销售策略。针对编号为 2 的客户（合资品牌），本文建议在原有基础上，对电池技术性能（电池耐用和充电方便）整体体验，提高一定的服务，比如在客户体验时，除了介绍合资品牌的自身亮点之外，相对强调一些在电池续航以及使用周期上的阐述，同时可以通过购车后几个月内免费充电等购车充电优惠，提高客户购车意愿概率。

针对编号为 5 的客户（自主品牌），结合最优组合以及原始各项服务满意度评分，

本文建议在提高动力性表现（爬坡和加速）服务的同时，相应的提高安全性表现（刹车和行车视野）服务，在提升服务之前，编号为 5 的客户在动力性表现的满意度评分上是各项服务中最高的，客户对爬坡和加速的体验更加青睐，所以对编号为 5 的客户进一步说明自主品牌的电动汽车动力性的优越性，并且阐述在动力性提高的同时，安全性能的优势。

针对编号为 15 的客户（新势力品牌），本文提出了两种销售策略。其一，对目标客户加强在安全性表现的服务上，通过具体的制动与刹车系统、安全性附加配置等的讲解，使用户提高新势力品牌的电动汽车在刹车与行车视野上的认识。并提出一些购车优惠性的服务，如购车即可免费拥有一次安全性能维护机会。其二，在安全性表现与驾驶操控性表现（转弯和高速的稳定性），两种项目上提高服务水平。例如，提高客户在新势力电动汽车的驾驶体验次数，并且阐述在各项道路场景中，驾驶舒适度、转弯流畅性等具体体现。同时在用户体验过程中，向客户阐述新势力电动汽车的安全性表现。

九. 问题五销售策略建议

贵司销售部门：

您好！

根据我们的研究成果，我们提出一些提高销售的建议：

一．关于合资品牌新能源电动车的建议：

根据我们的研究，目标客户更加看重合资品牌的电池技术性能和性价比，销售部门可以详细介绍电池的优良性能，如电池耐用和充电方便等性能，还可以推出购车一定期间内提供免费充电等优惠，以提高客户的购买意愿。

二．关于自主品牌新能源电动车的建议：

我们发现客户对自主品牌的动力性表现（爬坡和加速）以及安全表现（刹车和行车视野）十分的重视，销售部门可以通过广告等途径宣传该品牌的安全性和动力优越性，并与保险公司合作，推出购买该品牌电动车赠送保险服务等一系列优惠政策。

三．关于新势力品牌新能源电动车的建议：

我们研究发现新势力品牌的目标客户除了看重安全性，机动性之外，还十分看驾驶操控性表现。因此，我们建议在自主品牌电动车的销售策略基础上，推出客户免费试驾体验服务，还可以赠送电动车坐垫、把手套等提高客户舒适度的附属产品来吸引客户。

十. 模型的评价与推广

10.1 模型的评价

10.1.1 模型的优点

（1）考虑到数据的不平衡性，采用 ADASYN 采样，使得数据更加平衡

（2）在做特征筛选时，对不同类别的数据进行不同方法的特征筛选，对同一类别数据，经过不同方法的多层筛选，最后得出的影响的因素可信度得到有效的保证

（3）在问题三预测 15 名客户购买电动车的可能性时，运用了随机森林、LightGBM、Catboost、XGBoost 等多种算法，对各个客户选取最优的算法，使得预测更加精确。

（4）问题二运用的多层特征筛选可应用到影响因子较多的坏境。

10.1.2 模型的缺点

（1）对于问题四的处理，我们先通过了网格点搜索的方法找到了所有的可能策略，但网格搜索的运算量较大，可以进一步优化。

参考文献

[1]欧阳明高.我国节能与新能源汽车发展战略与对策[J].汽车工程,2006(04):317-321.

[2]张晓宇,赵海斌,周小柯.中国新能源汽车产业发展现状及其问题分析——基于我国汽车产业可持续发展的视角[J].理论与现代化,2011(02):60-66.

[3]Muth S Q,Potterat J J,Rothenberg R B. Birds of a feather: using a rotational box plot to assess ascertainment bias.[J]. International journal of epidemiology,2000,29(5):

[4]司守奎,孙兆亮.数学建模算法与应用[M].北京:国防工业出版社,2020.

[5]陈辉蓉.两总体的 wilcoxon 秩检验与 Kruskal-walls 检验的关系[J].数理医药学杂志,2000(03):213.

[6]赵婷,华一新,李响,李翔,杨飞.一种基于 Heat Map 的地理标签数据可视化表达的研究[J].测绘工程,2016,25(06):28-32.

[7]Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 1322-1328.

[8]Petr Somol,Jana Novovicová,Pavel Pudil. Notes on the evolution of feature selection methodology[J]. Kybernetika,2007,43(5):

[9]梁亚声,徐欣等.数据挖掘原理、算法与应用[M].北京:机械工业出版社,2014.11

[10]吴辰文,梁靖涵,王伟,李长生.基于递归特征消除方法的随机森林算法[J].统计与决策,2017(21):60-63.

[11]XGBoost:A Scalable Tree Boosting System. Chen T,Guestrin C.2016

[12]Light GBM:A Highly Efficient Gradient Boosting Decision Tree. KE G L,MENG Q,FINLEY T,et al. Advances in Neural Information Processing Systems 30 . 2017

[13]CatBoost:unbiased boosting with categorical features. L.Prokhorenkova. Advances in Neural Information Processing Systems . 2018

[14]周志华.机器学习[M].北京:清华大学出版社,2016

[15]范诗语,耿子悦,田芮绮,杜永强.基于集成学习的上市企业违约风险评价[J].统计与管理,2021,36(02):62-68.

[16]Park Kyeong mo. A Genetic-Based Optimization Model for Clustered Node Allocation System in a Distributed Environment[J]. The KIPS Transactions:PartA,2003,10A(1):

[17]Priyadarshini Ishaani,Cotton Chase. A novel LSTM-CNN-grid search-based deep neural network for sentiment analysis.[J]. The Journal of supercomputing,2021:

[18]王益成.试论企业标准体系的重建与评价[J].中国标准化,2003(02):30-31.

附录

1 .数据预处理　　\R 4.0.5

```
rm(list=ls())
setwd("C:/Users/59956/Desktop/Model")
library(openxlsx)
library(mice)
library(dplyr)

## import appendix 1
dat1 = read.xlsx("附录 1  目标客户体验数据.xlsx",sheet = 1)
rownames(dat1) = dat1[,1]
dat1 = dat1[,-1]
colnames(dat1)[1] = "type"
colnames(dat1)[ncol(dat1)]="label"
dat1[grep(1,dat1$type),"type"]="Type 1"
dat1[grep(2,dat1$type),"type"]="Type 2"
dat1[grep(3,dat1$type),"type"]="Type 3"
dat1$type = as.factor(dat1$type)
```

2. 异常值箱线图　　\R 4.0.5

```
pdf("Boxplots.pdf",height = 2.5*5,width = 2.5*8)
colors = brewer.pal(8,"Accent")
par(mfrow=c(5,8))
for (i in colnames(datn1)) {
#i=colnames(datn1)[1]
   boxplot(as.numeric(as.matrix(datn2[i])),col = "white",cex=0.5, pch=20,outcol=colors[5],
           main = i,ylab="Value",xlab="")
   stripchart(as.numeric(as.matrix(datn1[i])),vertical = T,method = "jitter",cex = 0.8,
              pch = 20,col = colors[5],add=T)
   boxplot(as.numeric(as.matrix(datn1[i])),col = "white",cex=0.5, pch=20,outcol=colors[6],
           main = i,ylab="Value",xlab="")
   stripchart(as.numeric(as.matrix(datn1[i])),vertical = T,method = "jitter",cex = 0.8,
              pch = 20,col = colors[6],add=T)
   }
dev.off()
```

3. 缺失值处理　　\python3.0

```
#加载所有常用模块
from pyforest import *
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv("F:/data/null.csv",encoding='gbk')
data.head()
```

```
data.shape
import missingno as msno
# 可视化看下缺省值
msno.matrix(data, labels = True)
msno.bar(data)
```

4. 归一化 \R 4.0.5

```
normalize = function(x) {
   return((x - min(x)) / (max(x) - min(x)))
}
datn = as.data.frame(lapply(dat1[,c(paste0("a",1:8),"B2","B4","B5","B7","B8","B10",
                                    paste0("B",13:17))], normalize))
```

5. 热力图 \R 4.0.5

```
library(corrplot)
pdf("ExampleHeatmap.pdf",height = 2.5*5,width = 2.5*5)
Cor    = cor(datn,method = "spearman")
corrplot:: corrplot(corr = Cor, method = "shade",type = "upper", tl.pos = "lt")
corrplot::corrplot(corr = Cor,add = TRUE, type = "lower", method = "number", diag = FALSE,
tl.pos = "n", cl.pos="n")
dev.off()
```

6. 小提琴图 \R 4.0.5

```
## draw violinplot
library(RColorBrewer)
p = list()
my_comparisons <- list( c(1, 2), c(2, 3), c(1, 3) )
color = c("#33B44A","#EE3536","#3A429B")
for (i in colnames(dat1)[2:9]) {

   p[[i]]=dat1 %>% ggplot(aes_string(x="type",y=i,fill="type"))+geom_violin()+
      geom_boxplot(width=0.2,position=position_dodge(0.9),outlier.colour                    =
NA,fill="white")+
      theme_bw()+
      theme(legend.position="none",
            axis.text.x = element_text(hjust = 0.5, vjust = 0.5),
            axis.text.y = element_text(hjust = 0.5, vjust = 0.5),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            axis.title.x = element_blank(),
            axis.title.y = element_text( size=rel(1)),
            panel.border = element_blank(),axis.line = element_line(colour = "black",size=1)
      )+
      scale_fill_manual(values = color)+
```

```
    coord_cartesian( ylim=c(0,130))+
    stat_compare_means(
                                comparisons = my_comparisons)+
    stat_compare_means(label.y = 130)
}

plot_grid(plotlist = p,align = "h",labels =    LETTERS[1:8])
ggsave("violin plots.pdf",height = 15,width = 15)
```

7. 树形图　　\python3.0

```
import squarify
#分类变量可视化
data=pd.read_csv("F:/data/B3.csv",encoding='gbk')
data.head()
#图片显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] =False #减号 unicode 编码
df = data.groupby('居住区域').size().reset_index(name='counts')
labels = df.apply(lambda x: str(x[0]) + "\n (" + str(x[1]) + ")", axis=1)
sizes = df['counts'].values.tolist()
colors = [plt.cm.Spectral(i/float(len(labels))) for i in range(len(labels))]
# Draw Plot
plt.figure(figsize=(12,8), dpi= 80)
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=.8)

# Decorate
plt.title('目标客户居住区域分布情况')
plt.axis('off')
plt.show()
```

8. 卡方检验　　\python3.0

```
data=pd.read_csv("F:/data/category3.csv",encoding='gbk')
data.head()
data.shape
import matplotlib.pyplot as plt#加载所有常用模块
from pyforest import *
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
X=data.drop(['label'], axis=1) #删除列
y=data['label']
from imblearn.over_sampling import ADASYN
X_resampled, y_resampled = ADASYN().fit_sample(X, y)
```

```
print('采样结果为：')
print(y_resampled.value_counts())
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
model1 = SelectKBest(chi2,k=4)#选择 k 个最佳特征
model1.fit_transform(X_resampled, y_resampled) #该函数可以选择出 k 个特征
model1.scores_  #得分
model1.pvalues_  #p-values
d = pd.DataFrame(model1.scores_) #转为 dataframe
d.to_csv("F:/data/chi.csv",index=False)#转换成 csv 文件
```

9. 最大信息系数法 \python3.0

```
import numpy as np
from sklearn.feature_selection import SelectKBest
from minepy import MINE
data=pd.read_csv("F:/data/info3.csv",encoding='gbk')
data.head()
data.shape
a=data.drop(['label'], axis=1) #删除列
b=data['label']
from imblearn.over_sampling import ADASYN
X_resampled, y_resampled = ADASYN().fit_sample(a, b)
print('采样结果为：')
print(y_resampled.value_counts())
#由于 MINE 的设计不是函数式的，定义 mic 方法将其为函数式的，返回一个二元组，
    二元组的第 2 项设置成固定的 P 值 0.5
def mic(x, y):
    m = MINE()
    m.compute_score(x, y)
    return (m.mic(), 0.5)
#由于 MINE 的设计不是函数式的，定义 mic 方法将其为函数式的，返回一个二元组，
    二元组的第 2 项设置成固定的 P 值 0.5
def mic(x, y):
    m = MINE()
    m.compute_score(x, y)
    return (m.mic(), 0.5)

#选择 K 个最好的特征，返回特征选择后的数据
new=SelectKBest(
    lambda X, Y: np.array(list(map(lambda x: mic(x, Y), X.T))).T[0],
    k=2).fit_transform(X_resampled, y_resampled)
new
```

10. 方差过滤法 \python3.0

```
data=pd.read_csv("F:/data/num1.csv",encoding='gbk')
data.head()
data=data.drop(['label'], axis=1) #删除列
data.shape
data.describe()
des1 = data.describe()
std1 = des1.loc['std'].sort_values(ascending=False)
X_16_1 = data[std1[std1>0.14].index] #设置方差阈值=0.1
X_16_1.shape
X_16_1.to_csv("F:/data/var.csv",index=True,encoding='gbk')#转换成 csv 文件
```

11. 随机森林度量特征重要性 \python3.0

```
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectFromModel
data=pd.read_csv("F:/data/ran3.csv",encoding='gbk')
data.head()
data.shape
X=data.drop(['label'], axis=1) #删除列
y=data['label']
#采样
from imblearn.over_sampling import ADASYN
X_resampled, y_resampled = ADASYN().fit_sample(X, y)
print('采样结果为：')
print(y_resampled.value_counts())
#基于随机森林度量各个变量的重要性
clf = RandomForestClassifier()
clf = clf.fit(X_resampled, y_resampled)
clf
importance1 = np.mean([tree.feature_importances_ for tree in clf.estimators_], axis=0)
std1 = np.std([tree.feature_importances_ for tree in clf.estimators_], axis=0)
indices = np.argsort(-importance1)  # 返回由大到小的数据之前的索引
indices
choose_num = 13  # 选择前 13 个因子
suoyin = indices[0:choose_num][::-1]
range_ = range(choose_num)
columns=X.columns #查看列名
feature_name = pd.Series(columns)
feature_name[suoyin]
#图片显示中文
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] =False #减号 unicode 编码
##画图
plt.figure(figsize=(12, 9))
```

```
plt.barh(range_, importance1[suoyin], color='r', xerr=std1[suoyin], alpha=1, align='center')
plt.xticks(fontsize=20)
plt.yticks(range(choose_num), feature_name[suoyin],fontsize=20)
plt.ylim([-1, choose_num])
# plt.xlim([0.0,0.055])
plt.xlabel('Feature importances',fontsize=20)
plt.show()
sum(importance1[suoyin])
```

12. RBF 递归特征消除 \python3.0

```
data=pd.read_csv("F:/data/tree3.csv",encoding='gbk')
data.head()
X = data.drop(['label'],axis=1) #从数据集中删除目标变量
y = data["label"]
#采样
from imblearn.over_sampling import ADASYN
X_resampled, y_resampled = ADASYN().fit_sample(X, y)
print('采样结果为：')
print(y_resampled.value_counts())
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

#递归特征消除法，返回特征选择后的数据
#参数 estimator 为基模型
#参数 n_features_to_select 为选择的特征个数
rfe=RFE(estimator=GradientBoostingClassifier()).fit_transform(X_resampled, y_resampled)
da = pd.DataFrame(rfe) #转为 dataframe
da.head()
da.to_csv("F:/data/RBF.csv",index=False)#转换成 csv 文件
```

13. 第三问代码 \python3.0

```
#压缩数据
def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the data type
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum()
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
```

```
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        else:
            df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum()
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))
    return df
sample_feature = reduce_mem_usage(pd.read_csv("F:/data/type11.csv",encoding='gbk'))
data=sample_feature
data.drop(['num'], axis=1, inplace=True)
data.head()
X=data.drop(['label'], axis=1) #删除列
y=data['label']
#划分训练测试集
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=33)
#采样
from imblearn.over_sampling import ADASYN
X_resampled, y_resampled = ADASYN().fit_sample(X_train,y_train)
print('采样结果为：')
print(y_resampled.value_counts())
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from bayes_opt import BayesianOptimization
```

14. 随机森林　　\python3.0

```
import xgboost as xgb
from bayes_opt import BayesianOptimization
#（1）随机森林
def rf_cv(n_estimators, min_samples_split, max_features, max_depth):
    rf=RandomForestClassifier(n_estimators=int(n_estimators),
                min_samples_split=int(min_samples_split),
                max_features=min(max_features, 0.999), # float
                max_depth=int(max_depth),
                random_state=2
            )
    rf.fit(X_resampled,y_resampled)
    y_pred=rf.predict(X_test)
    return metrics.f1_score(y_test, y_pred)

rf_bo = BayesianOptimization(
            rf_cv,
            {'n_estimators': (30, 250),
            'min_samples_split': (2, 25),
            'max_features': (0.1, 0.999),
            'max_depth': (5, 15)}
    )
rf_bo.maximize()
rf_bo.max
col=['target']
col.extend(list(rf_bo.max['params'].keys()))
C=[]
for can in rf_bo.res:
    target=can['target']
    par=can['params']
    c0=[target]
    c0.extend([int(par[k]) if k in ['n_estimators','min_samples_split','max_depth'] else
    round(par[k],4) for k in list(par.keys())])
    print(c0)
    C.append(c0)
rf_bys=pd.DataFrame(C,columns=col)
rf_bys.to_excel('F:/data/随机森林贝叶斯.xls')
clf1 = RandomForestClassifier(n_estimators=70, max_depth=15, min_samples_split=9,
                                max_features=0.1)
clf1.fit(X_resampled,y_resampled)
y_pred=clf1.predict(X_test)
from sklearn.metrics import accuracy_score
print('auc_score:{}'.format(accuracy_score(y_test, y_pred)))
```

```
print('Precision',metrics.precision_score(y_test, y_pred))
print('Recall',metrics.recall_score(y_test, y_pred))
print('F1-score:',metrics.f1_score(y_test, y_pred))
#预测每个类别的概率，这是叶子中相同类别的训练样本的分数
result=clf1.predict_proba(X_test)
result = pd.DataFrame(result)
result.head()
```

15. XGBoost \python3.0

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '1'
#（3）xgb
#需提前将自变量：X_jn 因变量 y 准备好
from sklearn.model_selection import RepeatedKFold
def xgb_cv(gamma,max_depth,lambda0,subsample,colsample_bytree,min_child_weight
           ,eta,num_rounds,n_estimator,reg_alpha):
    params = {
    'booster': 'gbtree',
    'objective': 'multi:softmax',  # 多分类的问题
    #'objective': 'reg:logistic',  # 多分类的问题
    'num_class': 2,                # 类别数，与 multisoftmax 并用
    'gamma': min(gamma,0.39),                    # 用于控制是否后剪枝的参数,越大越保守，一般 0.1、0.2 这样子。
    'max_depth': int(max_depth),                 # 构建树的深度，越大越容易过拟合
    'lambda': lambda0,                  # 控制模型复杂度的权重值的 L2 正则化项参数，参数越大，模型越不容易过拟合。
    'subsample': subsample,             # 随机采样训练样本
    'colsample_bytree':min(colsample_bytree,0.9),       # 生成树时进行的列采样
    'min_child_weight': int(min_child_weight),
    'n_estimator':int(n_estimator),
    'reg_alpha':reg_alpha,
    'silent': 0,                        # 设置成 1 则没有运行信息输出，最好是设置为 0.
    'eta': eta,                         # 如同学习率
    'seed': 1000,
    'nthread': 6,                        # cpu 线程数
    }
    F1=[]
    kf = RepeatedKFold(n_splits=4, n_repeats=2, random_state=10)
    for train_index, test_index in kf.split(x):
        X_train, y_train = x.loc[train_index], y.loc[train_index]
        X_test, y_test = x.loc[test_index], y.loc[test_index]
        X_resampled, y_resampled = ADASYN().fit_sample(X_train, y_train)  # 在训练集采样
        plst = params.items()
```

```
        dtrain = xgb.DMatrix(X_resampled, label=y_resampled) # 组成训练样本
        model = xgb.train(plst, dtrain, int(num_rounds)) # xgboost 模型训练
        test = xgb.DMatrix(X_test)
        test_pred=model.predict(test)
        F1.append(metrics.f1_score(y_test, test_pred))
    return np.mean(F1)
    xgb_bo = BayesianOptimization(
            xgb_cv,
            {'gamma': (0.05, 0.39),
            'max_depth': (3, 18),
            'lambda0': (0.05,2),
            'subsample': (0.4, 0.95),
            'colsample_bytree': (0.3, 0.8),
            'min_child_weight': (2, 8),
            'eta': (0.005, 0.09),
            'min_child_weight': (2, 8),
            'num_rounds':(5,13),
            'n_estimator':(50,300),
            'reg_alpha':(0.001,0.95)
            }
        )
    xgb_bo.maximize()

    xgb_bo.max
    xgb_bo.res
    xgb_bo.max
    #最优模型保存参数-存储贝叶斯过程数据

    params = {
        'booster': 'gbtree',
        'objective': 'multi:softmax',  # 多分类的问题
        'num_class': 2,                # 类别数，与 multisoftmax 并用
        'gamma': 0.36,                 # 用于控制是否后剪枝的参数,越大越保守，
一般 0.1、0.2 这样子。
        'max_depth': 16,               # 构建树的深度，越大越容易过拟合
        'lambda': 0.21,                # 控制模型复杂度的权重值的 L2 正则化项
参数，参数越大，模型越不容易过拟合。
        'subsample': 0.58,             # 随机采样训练样本
        'colsample_bytree': 0.33,      # 生成树时进行的列采样
        'min_child_weight': 2.4,
        'silent':  0,                  # 设置成 1 则没有运行信息输出，最好是设置
为 0.
        'eta': 0.06,                   # 如同学习率
        'seed': 1000,
```

```
        'nthread': 4,                          # cpu 线程数
    }
    xgb_max=xgb_bo.max['params']
    #将 xgb.max 转为模型可以使用的参数
    def get_params(params,xgb_max):
      xgb_max=xgb_bo.max['params']
      for k,v in zip(xgb_max.keys(),xgb_max.values()):
        if k!='num_rounds':
            #print(k,v)
            if k=='lambda0':
                params['lambda']=int(v)
            elif k in ['max_depth','min_child_weight','n_estimator']:
                params[k]=int(v)
            else:
                params[k]=v
        else:
            num_rounds = int(v)
      return params,num_rounds

    p0,n0=get_params(params,xgb_max)
    print(p0,n0)
    #将贝叶斯过程表存储
    col=['target']
    col.extend(list(xgb_max.keys()))
    C=[]
    for can in xgb_bo.res:
        target=can['target']
        par=can['params']
        c0=[target]
        c0.extend([int(par[k]) if k in ['max_depth','min_child_weight','lambda0','num_rounds'] else round(par[k],4) for k in list(par.keys())])
        print(c0)
        C.append(c0)
    xg_bys=pd.DataFrame(C,columns=col)
    xg_bys.to_excel('F:/data/xgb 贝叶斯.xls')
    from xgboost import XGBClassifier
    clf2 = XGBClassifier(booster= 'gbtree', objective='multi:softmax',num_class=2,
    gamma=0.11696860130369573, max_depth=8, reg_lambda=0, subsample=
0.8655511495012012,
     colsample_bytree= 0.40700451581623387, min_child_weight= 2, silent= 0,
     eta=0.05427410755931234, seed=1000, nthread= 4, n_estimator= 255,
     reg_alpha=0.19559357671270297,num_rounds=12)
    clf2.fit(X_resampled,y_resampled)
```

```
y_pred=clf2.predict(X_test)
from sklearn.metrics import accuracy_score
print('auc_score:{}'.format(accuracy_score(y_test, y_pred)))
print('Precision',metrics.precision_score(y_test, y_pred))
print('Recall',metrics.recall_score(y_test, y_pred))
print('F1-score:',metrics.f1_score(y_test, y_pred))
```

16. LightGBM \python3.0

```
from bayes_opt import BayesianOptimization
from lightgbm import LGBMClassifier
import lightgbm as lgb
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.metrics import roc_auc_score, accuracy_score
def lgb_cv(n_estimators, min_samples_split, max_features, max_depth):
    lgb=LGBMClassifier(n_estimators=int(n_estimators),
            min_samples_split=int(min_samples_split),
            max_features=min(max_features, 0.999), # float
            max_depth=int(max_depth),
            random_state=2
        )
    lgb.fit(X_resampled,y_resampled)
    y_pred=lgb.predict(X_test)
    return metrics.f1_score(y_test, y_pred)

lgb_bo = BayesianOptimization(
        lgb_cv,
        {'n_estimators': (30, 250),
        'min_samples_split': (2, 25),
        'max_features': (0.1, 0.999),
        'max_depth': (5, 15)}
    )
lgb_bo.maximize()
lgb_bo.max
gbm = LGBMClassifier(n_estimators=57, max_depth=14,
    min_samples_split=10.60807228663149,
                max_features=0.6829707285490627
gbm.fit(X_resampled,y_resampled)
from sklearn.metrics import accuracy_score
from sklearn import metrics
print('auc_score:{}'.format(accuracy_score(y_test, y_pred)))
print('Precision',metrics.precision_score(y_test, y_pred))
```

```
print('Recall',metrics.recall_score(y_test, y_pred))
print('F1-score:',metrics.f1_score(y_test, y_pred))
### lgb 效果最好，用来预测
data=pd.read_csv("F:/data/brand1.csv",encoding='gbk')
data.head()
X=data.drop(['num'], axis=1) #删除列
y_pred=gbm.predict(X)
y_pred
#预测每个类别的概率，这是叶子中相同类别的训练样本的分数
result=gbm.predict_proba(X)
result = pd.DataFrame(result)
result.head()
result.to_csv("F:/data/ypred.csv",index=False)#转换成 csv 文件
data=pd.read_csv("F:/data/ypred.csv",encoding='gbk')
data.head()
#查看概率分布
sns.distplot(data['1'])
plt.show()
```

17. CatBoost \python3.0

```
import pandas as pd, numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import metrics
import catboost as cb
import catboost as CatBoost
import pandas as pd
import numpy as np
from catboost import CatBoostClassifier, CatBoost, Pool, cv
from bayes_opt import BayesianOptimization
def cab_cv(depth, learning_rate, l2_leaf_reg, iterations):
    cab=cb.CatBoostClassifier()
    cab.fit(X_resampled,y_resampled)
    y_pred=cab.predict(X_test)
    return metrics.f1_score(y_test, y_pred)

cab_bo = BayesianOptimization(
        cab_cv,
        {'depth': (5,15),
         'learning_rate': (0.03,0.15),
         'l2_leaf_reg': (1,9),
         'iterations': (300, 500)}
    )
cab_bo.maximize()
cab_bo.max
```

```
#用最优参数拟合数据
clf = cb.CatBoostClassifier(depth=7, iterations=320, l2_leaf_reg=5.521070327796334,

    learning_rate=0.06522340040216315,loss_function="Logloss")
clf.fit(X_resampled,y_resampled)
y_pred=clf.predict(X_test)
from sklearn.metrics import accuracy_score
from sklearn import metrics
print('auc_score:{}'.format(accuracy_score(y_test, y_pred)))
print('Precision',metrics.precision_score(y_test, y_pred))
print('Recall',metrics.recall_score(y_test, y_pred))
print('F1-score:',metrics.f1_score(y_test, y_pred))
```

18. 满意度柱形图　　\R 4.0.5

```
dat3 = read.xlsx("附录 3  待判定的数据.xlsx",sheet = 1)
rownames(dat3) = dat3[,1]
dat3 = dat3[,-1]
colnames(dat3)[1] = "type"
dat3[grep(1,dat3$type),"type"]="Type 1"
dat3[grep(2,dat3$type),"type"]="Type 2"
dat3[grep(3,dat3$type),"type"]="Type 3"
dat3$type = as.factor(dat3$type)
dat3 = dat3[c(2,8,15),]

## draw barplot
p = list()
color = c("#33B44A","#EE3536","#3A429B")
for (i in colnames(dat3)[2:9]) {
 # i=colnames(dat3)[2]
  p[[i]]=dat3 %>% ggplot(aes_string(x="type",y=i,fill="type"))+geom_bar(stat =
    "identity",width =0.6)+
    geom_text(aes(label = dat3[,i], vjust = -0.8, hjust = 0.5), show.legend = TRUE)+
    theme_bw()+
    theme(legend.position="none",
          axis.text.x = element_text(hjust = 0.5, vjust = 0.5),
          axis.text.y = element_text(hjust = 0.5, vjust = 0.5),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_text( size=rel(1)),
          plot.title=element_text(size=rel(1.5),hjust = 0.5),
          panel.border = element_blank(),axis.line = element_line(colour = "black",size=1)
    )+
    scale_fill_manual(values = color)+labs(title =i)+ylab("Value")+
```

```
      coord_cartesian( ylim=c(65,90))
}

plot_grid(plotlist = p,align = "h",labels =    LETTERS[1:8])
ggsave("barplots.pdf",height = 15,width = 15)
```

19. 网格搜索　　\R 4.0.5

```
library(e1071)
TestSample1 =c (85.04342101,81.55817068,2,9,10,10)
RepeatSample1 = matrix(rep(TestSample1,36),nrow = 36,ncol = 6,byrow = T)
Cmatrix = expand.grid(0:5,0:5)
Szero = matrix(rep(0,36*4),nrow = 36,ncol=4)
Advance = cbind(Cmatrix,Szero)
TestSampleMat1 = RepeatSample1 +Advance
colnames(TestSampleMat1) = c("a1","a3","B3","B12","B16","B17")
write.csv(TestSampleMat1,file = "TestSample1.csv",row.names=F)


TestSample2 =c (82.79728523,85.6115933,2,2,30,10)
RepeatSample2 = matrix(rep(TestSample2,36),nrow = 36,ncol = 6,byrow = T)
TestSampleMat2 = RepeatSample2 +Advance
colnames(TestSampleMat2) = c("a1","a4","B3","B12","B16","B17")
write.csv(TestSampleMat2,file = "TestSample2.csv",row.names=F)


TestSample3 =c (81.09094276,70.44455435,80.84886951,5,6,0)
RepeatSample3 = matrix(rep(TestSample3,6^3),nrow = 6^3,ncol = 6,byrow = T)
Cmatrix = expand.grid(0:5,0:5,0:5)
Szero = matrix(rep(0,216*3),nrow = 216,ncol=3)
Advance = cbind(Cmatrix,Szero)
TestSampleMat3 = RepeatSample3 +Advance
colnames(TestSampleMat3) = c("a2","a3","a6","B11","B12","B16")
write.csv(TestSampleMat3,file = "TestSample3.csv",row.names=F)

## put theme into the lightGBM model to get the optimal combination
```